AUTOMATIC RED EYE REMOVAL

Donghui Wu

FIELD OF INVENTION

[0001] This invention relates to a method for red eye removal in photographs and images.

DESCRIPTION OF RELATED ART

[0002] The pupil is an opening that lets light into the eye. Since most of the light entering the eye does not escape, the pupil appears black. In dim light, the pupil expands to allow more light to enter the eye. In bright light, the pupil contracts to allow less light to enter the eye.

[0003] "Red eye" is a phenomenon where a person's pupils appear red in a photograph taken with a flash. Red eye comes from light reflecting off of the blood vessels in the retinas (on the back interior of the eyeballs).

[0004] Some cameras have a "red eye reduction" feature. In these cameras, the flash goes off twice -- once right before the picture is taken, and then again to actually take the picture. The first flash causes people's pupils to contract, thereby reducing red eye significantly.

[0005] Some photography software have a "red eye removal" feature. These software require the user to identify the red eye, usually by dragging a rectangular box around the red eye with a mouse, and then remove red eye from the specified area. Others software, such as those available from Pixology of Guildford, England, and FotoNation of San Francisco, California, require little or no user intervention and the software identifies the red eye automatically.

[0006] In addition, there are literatures that describe methods for red eye removal. Of particular interest is U.S. Patent Application Publication 2002/0176623, filed March 29, 2002 ("Steinberg"). Steinberg describes a 3-step process: (1) search candidate areas by color constraints; (2) eliminate candidates with shape and other criteria; and (3) output the results for user interactive verification of the red eye candidates. However, this process has been widely used and is processing common sense.

[0007] For example, in an article by de la Escalera et al., a process for detecting road traffic signs is disclosed. de la Escalera et al., "Road Traffic Sign Detection and Classification," IEEE Trans. on Industrial Electronics, Vol. 44, No. 6, Dec. 1997. de la Escalera et al. discloses two steps: (1) localize the sign in the image depending on the color and the form; and (2) recognize the sign through a neural network.

[0008] In another example, in an article by Yang et al., a process for face detection is disclosed. "Detecting Faces in Images; A Survey," Yang et al., IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 24, No. 1, Jan. 2002. Yang et al. states, "Most of [software] utilize global features such as skin color, size, and shape to find face candidates, and then verify these candidates using local, detailed features such as eye brows, nose, and hair. A typical approach begins with the detection of skin-like regions .... Next, skin-like pixels are grouped together using connected component analysis or clustering algorithms. If the shape of a connected region has an elliptic or oval shape, it becomes a face candidate. Finally, local features are used for verification." Id. at p. 40.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] Fig. 1 illustrates an image with red eye in one embodiment of the invention.

[0010] Fig. 2 is a flowchart of a method for red eye removal in one embodiment of the invention.

[0011] Figs. 3, 4, and 5 illustrate how a red eye region is determined to be a substantially round pupil with red eye in one embodiment of the invention.

[0012] Fig. 6 illustrates how a red eye region is determined to be too close to another red eye region in one embodiment of the invention.

[0013] Figs. 7 and 8 illustrate how a red eye region is determined to be proximate to a facial region in one embodiment of the invention.

[0014] Figs. 9 and 10 illustrate how a red eye region is determined to be proximate to a white of an eye in one embodiment of the invention.

[0015] Use of the same reference numbers in different figures indicates similar or identical elements.

SUMMARY

[0016] In one embodiment of the invention, a method for removing a red eye from an image includes (1) calculating a weighted red value for each pixel in the image from red, green, and blue color values and a luminance value of each pixel in the image, (2) selecting a plurality of pixels in the image having weighted red values greater than a threshold as red eye pixels, and (3) correcting some of the red eye pixels to remove the red eye from the image. The weighted red value for a pixel is calculated as follows: $f = \frac{c_1 r + c_2 g + c_3 b}{Y}$, wherein f is the weighted red value, r is the red color value, g is the green color value, b is the blue color value, c_1 is a first weight given to the red color value, c_2 is a second weigh given to the green color value, c_3 is a third weight given to the blue color value, and Y is the luminance.

DETAILED DESCRIPTION

[0017] Fig. 1 illustrates an image 10 of a person 12 having pupils 14, eyes 16, scleras 18, and face 20 in one embodiment. In a phenomenon called "red eye," pupils 14 appear red when light reflects off of the blood vessels in back interior of the eyeballs. It has been experimentally determined that a red eye typically consists of pixels that are purple-red or orange-red.

[0018] Fig. 2 illustrates a method 100 to automatically remove red eye from image 10 in one embodiment. Method 100 does not require the user to designate an area from which the red eye is removed. Method 100 involves low computational complexity and may be implemented in software executed by a computer or firmware embedded into digital cameras, printers, scanners, mobile phones.

[0019] In step 102, the software determines a weighted purple-red value of each pixel in image 10. The weighted purple-red value represents the likelihood that a pixel is a purple-red pixel that forms part of a red eye. In one embodiment for the rgb (red, green, and blue) color space, the weighted purple-red value of a pixel is determined from its red, green, and blue color values, and

its luminance as follows:

$$f_1 = \frac{c_1^{(1)} r + c_2^{(1)} g + c_3^{(1)} b}{Y}, \quad (1)$$

where f_1 is the weighted purple-red value, r is the red color value, g is the green color value, b is the blue color value, $c_1^{(1)}$ is a weight given to the red color value, $c_2^{(1)}$ is a weight given to the green color value, $c_3^{(1)}$ is a weight given to the blue color value, and Y is the luminance calculated as $Y = a_1 r + a_2 g + a_3 b$. Note that the weighted red value is independent of luminance change. In one embodiment, $c_1^{(1)}$ is 0.5, $c_2^{(1)}$ is 0.5, $c_3^{(1)}$ is -1, and $Y = 0.299r + 0.587g + 0.114b$. In other words, equation (1) can be rewritten as:

$$f_1 = \frac{r + b - 2g}{2Y}. \quad (2)$$

[0020] In one embodiment for the YCrCb (luminance, red chrominance, and blue chrominance) color space, the weighted purple-red value of a pixel is determined as follows:

$$f_1 = \frac{1.41514(Cr - 128) + 1.23014(Cb - 128)}{Y}, \quad (3)$$

where Cr is the red chrominance, Cb is the blue chrominance, and Y is the luminance.

[0021] In step 104, the software selects the pixels in image 10 that have weight purple-red values greater than a threshold as redeye pixels. In one embodiment, this threshold is 0.5.

[0022] In step 106, the software determines a weighted orange-red value of each pixel in image 10. The weighted orange-red value represents the likelihood that a pixel is an orange-red pixel that forms part of a red eye. In one embodiment, the weighted orange-red value for a pixel is determined from its red, green, and blue color values as follows:

$$f_2 = \frac{c_1^{(2)} r + c_2^{(2)} g + c_3^{(2)} b}{Y}, \quad (4)$$

where f_2 is the second type of weighted red value, $c_1^{(2)}$ is a weight given to the red color value, $c_2^{(2)}$ is a weight given to the green color value, and $c_3^{(2)}$ is a weight given to the blue color value. As noted above, the weighted red value is independent of luminance change. In one embodiment, $c_1^{(2)}$ is 0.6667, $c_2^{(2)}$ is 0.3333, $c_3^{(2)}$ is -1.0. In other words, equation (4) can be rewritten as:

$$f_2 = \frac{2r + g - 3b}{3Y}. \quad (5)$$

[0023] In one embodiment for the YCrCb color space, the weighted orange-red value of a pixel is determined as follows:

$$f_2 = \frac{0.69662(Cr - 128) - 1.88671(Cb - 128)}{Y}. \quad (6)$$

[0024] In step 108, the software selects the pixels in image 10 that have weighted orange-red values greater than another threshold as red eye pixels. In one embodiment, this threshold is 0.5.

[0025] In step 110, the software groups contiguous red eye pixels into red eye regions. Red eye pixels hereafter refer to the purple-red and orange-red pixels selected in steps 104 and 108. A red eye region can contain purple-red pixels, orange-red pixels, or a combination thereof. Fig. 3 illustrates a group of red eye pixels 202 (only a few are labeled) that form a red eye region 204. Although not illustrated, it is understood that other pixels surround red eye pixels 202.

[0026] In step 112, the software determines if each red eye region is a substantially round pupil. In step 114, the software rejects the red eye pixels in each red eye region that is not a substantially round pupil. Steps 112 and 114 are illustrated in reference to red eye region 204 in Figs. 3 and 4 but in actually are repeated for each red eye region.

[0027] First, the software determines the geometric center 206 of red eye region 204. The software then determines the distance D_{max} from geometric center 206 to the farthest red eye pixel 202F. Distance D_{max} is used to set a range of radii of circles where the weighted red values of corresponding pixels located on two adjacent circles are compared. This is illustrated in reference to Fig. 4.

[0028] The software first determines the sum of the weighted red values of the pixels located at a first circle having a radius R_i as follows:

$$S(R_i) = \sum_{0 \le \theta < 360} Max(f_1(R_i, \theta), f_2(R_i, \theta)) , \quad (7)$$

where $S(R_i)$ is the sum of the first circle, $Max(x,y)$ is a function that outputs the maximum of inputs x and y, and R_i and θ are the polar coordinates of a pixel.

[0029] The software then determines of the sum of the weight red values of the pixels located at an adjacent second circle having a radius R_{i+1}.

$$S(R_{i+1}) = \sum_{0 \le \theta < 360} Max(f_1(R_{i+1}, \theta), f_2(R_{i+1}, \theta)) , \quad (8)$$

where $S(R_{i+1})$ is the sum of the second adjacent circle. In one embodiment, the radius R_{i+1} is raidus R_i incremented by 1 pixel, and angles θ consists 0 to 360° increased at 4° increments (e.g., 0, 4, 8 ... 356).

[0030] The software then determines the difference between the two sums: $S(R_i) - S(R_{i+1})$. If the absolute value of the difference is small, then there has not been a change in the red color between the pixels on the adjacent circles, which indicates that the image has not transitioned from a pupil having red eye to the eye (e.g., from pupil 14 to eye 16). If the difference is positive, then there has been a decrease in the red color between the pixels on the adjacent circles, which may indicate that the image has transitioned from a pupil having red eye to the eye.

[0031] In one embodiment, this process is repeated for radii ranging from $0.5*D_{max}$ to $1.5*D_{max}$ where the radius is incremented at 1 pixel between adjacent circles. After the process is completed for these radii, the software selects the radius of the circle that generates the largest difference with its adjacent circle as the radius of a pupil (hereafter "R_{pupil}") having red eye region 204.

[0032] Referring to Fig. 5, the software then determines a first ratio between (1) the number of red eye pixels located within a circle 222 having radius of R_{pupil} and (2) the area of circle 222 in pixels as follows:

$$R_1 = \frac{N_{R_{pupil}}}{A_{R_{pupil}}}, \tag{9}$$

where R_1 is the first ratio, $N_{R_{pupil}}$ is the number of red eye pixels within a circle having radius R_{pupil}, and $A_{R_{pupil}}$ is the area of the circle.

[0033] The software also determines a second ratio between (1) the number of red eye pixels located within a ring 224 having an inner radius of R_{pupil} and an outer radius of D_{max} and (2) the area of ring 224 in pixels as follows:

$$R_2 = \frac{N_{R_{pupil}/D_{max}}}{A_{R_{pupil}/D_{max}}}, \tag{10}$$

where R_2 is the second ratio, $N_{R_{pupil}}$ is the number of red eye pixels within a ring having an inner radius R_{pupil} and an outer radius D_{max}, and $A_{R_{pupil}/D_{max}}$ is the area of the circle.

[0034] The software then determines a difference between the first ratio and the second ratio: R_1-R_2. A large difference indicates that red eye region 204 is probably a pupil with red eye. If the difference is less than a threshold, the software then rejects the red eye pixels in red eye region 204 as candidates for red eye removal. In one embodiment, the threshold is 30% (i.e., 0.3).

[0035] In step 116, the software determines if each remaining red eye region is too close to another red eye region. In step 118, the software rejects red eye pixels in each red eye region that is too close to another red eye region. Steps 116 and 118 are illustrated in reference to red eye regions 204 and 242 in Fig. 6 but in actually are repeated for each red eye region.

[0036] The software determines a distance D_{pupil} between the geometric centers of red eye regions 204 and 242. The software then determines if distance D_{pupil} is within a range that makes red eye regions 204 and 242 too close. In one embodiment, red eye regions are too close if they

are within $10*R_{pupil}$ to $14*R_{pupil}$ of each other. The software thus rejects the red eye pixels in red eye region 204 and 242 as candidates for red eye removal if they are too close to each other.

[0037] In step 120, the software determines if each remaining red eye region is proximate to a facial region (e.g., face 20 in Fig. 1). In step 122, the software rejects the red eye pixels in each red eye region that is not proximate to a facial region. Steps 120 and 122 are illustrated in reference to red eye region 204 in Figs. 7 and 8 but in actually are repeated for each red eye region.

[0038] The software first generates a histogram 262 of the pixels located in a ring 264 about geometric center 206. The function of the histogram is simply the number of pixels that has a particular color (e.g., a particular combination of Y,Cr,Cb color values). In one embodiment, ring 262 has an inner radius of $4*R_{pupil}$ and an outer radius of $9*R_{pupil}$. The software then selects the most common color 266 in histogram 262 and compares it to a range of threshold skin color values. If color 266 is within the range of threshold skin color values, then red eye region 204 is probably a pupil with red eye that is proximate to a facial region. In one embodiment, the threshold skin color values are expressed in HSV (hue, saturation, value) color space as - $80<H<50$, $5<S<80$, and $20<V<80$. Thus, the software first converts the most common color 266 into HSV color space and then compares it with the threshold skin color values. If color 266 is not within the range of threshold skin color values, the software rejects the pixels in red eye region 204 as candidates for red eye removal. In order to remove the luminance change, the luminance (Y) of the image within the circle having radius $9*R_{pupil}$ will be normalized to [0,255] before the software generates histogram 262 and compare color 266 to the range of threshold skin color values. This is because without the normalization, any luminance change will introduce a color cast (i.e., unwanted color effect) into the HSV color space.

[0039] In step 124, the software determines if each remaining red eye region is proximate to a sclera of an eye (e.g., sclera 18 in Fig. 1). In step 126, the software rejects the red eye pixels in each red eye region that is not proximate to a sclera of an eye. Steps 124 and 126 are illustrated in reference to red eye region 204 in Figs. 9 and 10 but in actually are repeated for each red eye region.

[0040] The software generates a luminance histogram 282 of the pixels located in a ring 284 about geometric center 206. In one embodiment, ring 284 has an inner radius of $2*R_{pupil}$ and an outer radius of $5*R_{pupil}$. From histogram 282, the software determines a ratio between (1) the number of pixels having the brightest color 288 in ring 284 and (2) the number of red eye pixels in a circle 286 having a radius of R_{pupil} as follows:

$$R_{sclera} = \frac{N_{brightest}}{N_{R_{pupil}}}, \tag{11}$$

where R_{sclera} is the ratio and $N_{brightest}$ is the number of pixels having the brightest color in ring 284.

[0041] If the ratio is greater than a sclera threshold, then red eye region 204 is probably a pupil with red eye proximate to the sclera. If the ratio is less than the threshold, then the software rejects the red eye pixels in red eye region 204. In one embodiment, the sclera threshold is 82% (e.g., 0.82).

[0042] In step 128, the software replaces the remaining red eye pixels with black pixels to remove the red eye.

[0043] The steps in the present invention are different from those in the conventional methods such as Steinberg described above. For example, Steinberg uses the conventional Lab color space but the current invention uses a specific color transform to discriminate the red eye pixels from the other pixels. Steinberg also uses an elongation ratio and a compaction ratio to disqualify candidate regions while the current invention uses round shape detection in terms of a polar coordinate system. Furthermore, the current invention also uses the distance between candidate regions to eliminate erroneous candidates, which is not disclosed by Steinberg.

[0044] Various other adaptations and combinations of features of the embodiments disclosed are within the scope of the invention. Numerous embodiments are encompassed by the following claims.